AMENDMENT TO THE SPECIFICATION

Please amend the specification as follows.

Please replace paragraph [80] with the following:

--By way of example, a path selection rule may select the path based on any of the following path information in which IP packets match the rule: a primary path, a secondary path, and a tertiary path. The primary path ~~is~~ may be specified in any path selection rule. The secondary path is used only when the primary path has failed. If no secondary path is specified, any IP packets that match the rule can be discarded when the primary path fails. The tertiary path is specified only if a secondary path is specified. The tertiary path is selected if both the primary and secondary paths have failed. If no tertiary path is specified, any IP packets that match the rule can be discarded when both the primary and secondary paths fail. Path selection may be generalized such that the path selection rule can select up to N paths where the Nth path is used only if the (N-1)th path fails. The example above where N=3 is merely illustrative, although N is typically a fairly small number.--

Please replace paragraph [94] with the following:

--When a Connection Request message is received from a peer TSK (step 403), the TCP Spoofing Kernel 280 allocates a CCB for the connection and then stores all of the relevant information from the CR message in the CCB. TSK 280 of PEP end point 404 then uses this information to generate a TCP <SYN> segment, as in step 415, to send to the remote host 406. The MSS in the <SYN> segment is set to the value received from the TSK peer of PEP end point 404. When the remote host responds with a TCP <SYN,ACK> segment (step 417), TSK 280 of PEP end point ~~402~~ 404 sends a Connection Established message to its TSK peer of the ~~remote~~ PEP end point ~~404~~ 402 (step 419), including in the CE message the MSS that is sent by the local host in the <SYN,ACK> segment. TSK 280 of PEP end point ~~402~~ 404 also responds, as in step 421, with a TCP <ACK> segment to complete the local three–way handshake. The peer TSK of PEP end point 404 then forwards the data that is received from TSK 280 to the host, per step 423. Concurrently, in step 425, the remote host 406 sends data to the peer TSK of PEP end point 404, which acknowledges receipt of the data by issuing an <ACK> segment to the ~~remote PEP~~

~~end point 404~~ host 406, per step 427. Simultaneously with the acknowledgement, the data is sent to TSK 280 of PEP end point 402 (step 429).--

Please replace paragraph [102] with the following:

-- For WAN-to-local traffic (i.e., downstream direction), the remote PEP end point 503 receives IP packets from its WAN interface 230 (Figure 2). IP packets that are not addressed to the end point 503 are simply forwarded (as appropriate) to the local interface 220 (Figure 2). IP packets addressed to the end point 503, which have a next protocol header type of "PEP Backbone Protocol (PBP)" are forwarded to the backbone protocol kernel 503c. The backbone protocol kernel 503c extracts the TCP data and forwards it to the TCP spoofing kernel 503b for transmission on the appropriate spoofed TCP connection. In addition to carrying TCP data, the backbone protocol connection is used by the TCP spoofing kernel 501b to send control information to its peer TCP spoofing kernel 503b in the remote PEP end point 503 to coordinate connection establishment and connection termination.--

Please replace paragraph [101] with the following:

-- Figure 6 illustrates the flow of IP packets through a PEP end point, according to an embodiment of the present invention. When IP packets are received at the local LAN interface 220, the PEP end point 210 determines (as shown by decision point A), whether the packets are destined for a host that is locally situated; if so, the IP packets are forwarded to the proper local LAN interface 220. If the IP packets are destined for a remote host, then the PEP end point 210 decides, per decision point B, whether the traffic is a TCP segment. If the PEP end point 210 determines that in fact the packets are TCP segments, then the TSK 280 determines whether the TCP connection should be spoofed (decision point C). However, if the PEP end point 210 determines that the packets are not TCP segments, then the BPK 282 processes the traffic, along with the PK 284 and the PSK 286 for eventual transmission out to the WAN. It should be noted that the BPK 282 does not process unspoofed IP packets; i.e., the packets flow directly to PK 284. As seen in Figure 6, traffic that is received from the WAN interface 230 is examined to determine whether the traffic is a proper PBP segment (decision point D) for the particular PEP end point 210; if the determination is in the affirmative, then the packets are sent to the BPK 282 and then the TSK 280.--

Please replace paragraph [112] with the following:

-- Figure 8 shows the interfaces of the PEP end point implemented as an IP gateway, according to one embodiment of the present invention. By way of example, an IP Gateway 801 has a single local LAN interface, which is an enterprise interface ~~803~~ 801. The IP Gateway 803 employs two WAN interfaces 805 for sending and receiving IP packets to and from remote site PEP End Points: a backbone LAN interface and a wide area access (WAA) LAN interface. --

Please replace paragraph [116] with the following:

-- Figure 10 shows a Multimedia VSAT implementation of the PEP end point, according to one embodiment of the present invention. A Multimedia VSAT 1001, in an exemplary embodiment, has two local LAN interfaces 1003. Support for one or more local PPP serial port interfaces may be utilized. The Multimedia VSAT 1001 has two WAN interfaces 1005 for sending IP packets to hub site PEP End Points: a VSAT inroute and one of its LAN interfaces. The Multimedia VSAT 1001 thus has three interfaces for receiving IP packets from hub site PEP End Points, the VSAT outroute and both of its LAN interfaces 1003. A Multimedia VSAT ~~1003~~ 1001 may support uses of both of its LAN interfaces ~~1003~~ 1001 at the same time for sending and receiving IP packets to and from hub site PEP End Points. The Multimedia VSAT 1003 further supports the use of a VADB serial port interface for sending and receiving IP packets to and from the hub site PEP End Points. --

Please replace paragraph [143] with the following:

-- On the surface, it might appear that changing either the PBP header 1525 or TSK header 1519 so that the combined headers equal 20 bytes to match the size of the TCP header 1515 may improve buffer handling performance (at the expense of wasting a couple of bytes of overhead when sending PBP segments across the WAN). However, in addition to reducing flexibility regarding handling TCP options, when looked at more closely, it is observed that this is not the case. The reason for this is that TSK and BPK buffer handling occur independently. TSK 280 is not aware of the size of the PBP header 1525 and should not be. And, conversely, BPK 282 is not aware of the size of the TSK header 1519 and should not be. Making the kernels aware of each other's header sizes violates their protocol layering relationship and would introduce an undesirable dependency between the kernels. The method defined to handle this problem is to use extra space at the front of the buffer along with a "pointer" (i.e., an offset count) to the buffer payload (e.g., the current start of the IP packet). This method allows the data to remain in place with only the buffer headers moved around. And, it takes advantage of the fact that the PEP kernels generally only reuse the space for headers. Fields in a header rarely remain unchanged and, therefore, a shift in the location of a header simply requires a change in the location where a kernel needs to fill in fields not an actual shift of header contents. For example, the IP header 1501 required by the TCP Spoofing Kernel 280 to send and receive TCP data to and from the local host contains no field values in common with the IP header 1501 required by the Backbone Protocol Kernel 282 to send and receive the same data across the WAN. And, the TCP header 1515 used to send and receive data to and from the local host is completely replaced by the PBP and TSK headers ~~1519~~ 1525 and ~~1509~~ 1519 used to send and receive the same data across the WAN (and vice versa). In an exemplary embodiment, in a buffer that has not had any header adjustments, the payload offset may point 44 bytes into the buffer at the start of an IP packet (because the buffer, in this example, is initialized with 16 bytes of header growth space). If a header needs to be inserted which is smaller than the header it is replacing, then the kernel which is making the adjustment moves the headers to the right, updating the payload field in the buffer. If a header needs to be inserted which is larger than the header it is replacing, then the kernel which is making the adjustment moves the headers to the left, again updating the payload offset field 1601b in the buffer. Of course, as indicated above, even when no header adjustments are required, payload offset adjustments may be required because IP packets are not the buffer "unit" passed at all interfaces. In particular, TSK messages are the buffer "unit" passed between the TCP Spoofing Kernel 280 and the Backbone Protocol Kernel 282.--

Please replace paragraph [170] with the following:

-- The following discussion describes the calculations that are performed by the platform environment 210, the TCP Spoofing Kernel 280, and the Backbone Protocol Kernel 282 to convert buffer space availability into advertised receive window sizes. For each backbone connection, as shown above, the platform environment 210 derives the amount of buffer space that can be used in the WAN to LAN direction for the connection by multiplying the per peer WAN to LAN buffer space value by the percentage of the per peer resources which have been allocated to this backbone connection. The resulting value is then used as the upper bound for WAN to LAN buffer space for this backbone connection. Because the per peer WAN to LAN buffer space values may be different in each peer, the platform environment 210 cannot directly calculate the corresponding limit for the amount of LAN to WAN buffer space even though the PEP End Point peers may share the same percentage of resources parameters; instead, this value is provided by the TCP Spoofing Kemel 280. The environment 210 provides the WAN to LAN buffer limit (and the local number of CCBs limit) to TSK 280 when it opens the backbone connection. TSK 280 then sends the limit to its TSK peer in a TSK Peer Parameters (TPP) message. When TSK 280 receives a TPP message, it extracts the peer's WAN to LAN buffer space limit from the message and passes it to the environment. The environment uses the peer's WAN to LAN buffer space limit as its local LAN to WAN buffer space limit. When a backbone connection is first opened, while waiting for the reception of a TPP message from the peer, the LAN to WAN buffer space limit and the peer number of CCBs limit are initialized to 0. This prevents TCP connections from being spoofed until valid peer parameter information is received. As described previously, the platform environment 210 counts the number of LAN to WAN buffers 1201 and WAN to LAN buffers 1203 it has allocated to each backbone connection in the backbone connection's ECB. When a buffer is allocated, the appropriate in use count is incremented. When a buffer is deallocated, the backbone connection handle stored by the environment in the buffer is used to find the proper in use count to decrement. When requested by TSK 280 or BPK 282, the platform environment 210 returns the currently available LAN to WAN or WAN to LAN buffer space for a backbone connection. In a platform (e.g. the PES Remote) where small, chained buffers are used, the platform environment 210 must normalize its buffer count based on the number of buffers required to hold a maximum size IP packet. TSK 280 and BPK 282 use these values to calculate window sizes, as follows:

$$A_{pi}^{W2L} = B_{pi}^{W2L} - U_{pi}^{W2L}$$

$$A_{pi}^{L2W} = B_{pi}^{L2W} - U_{pi}^{L2W},$$

where B_{pi}^{W2L} is the calculated WAN to LAN buffer space limit for the backbone connection to peer "p" at priority "i", B_{pi}^{L2W} is the learned LAN to WAN buffer space limit for the backbone connection to peer "p" at priority "i", U_{pi}^{W2L} is the WAN to LAN buffer space in use for the backbone connection to peer "p" at priority "i", U_{pi}^{L2W} is the LAN to WAN buffer space in use for the backbone connection to peer "p" at priority "i", A_{pi}^{W2L} is the WAN to LAN buffer space available for the backbone connection to peer "p" at priority "i", and A_{pi}^{L2W} is the LAN to WAN buffer space available for the backbone connection to peer "p" at priority "i".--

Please replace paragraph [183] with the following:

-- Network link 3021 typically provides data communication through one or more networks to other data devices. For example, network link 3021 may provide a connection through local area network 3023 to a host computer 3025 or to data equipment operated by an Internet Service Provider (ISP) 3027. ISP 3027 in turn provides data communication services through the Internet ~~505~~. In addition, LAN 3023 is linked to an intranet ~~3029~~. The intranet ~~3029~~, LAN 3023 and Internet ~~505~~ all use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 3021 and through communication interface 3019 which carry the digital data to and from computer system 3001, are exemplary forms of carrier waves transporting the information.--

Please replace paragraph [184] with the following:

-- Computer system 3001 can send messages and receive data, including program code, through the network(s), network link 3021 and communication interface 3019. In the Internet example, a server ~~3031~~ might transmit a requested code for an application program through Internet ~~505~~, ISP 3027, LAN 3023 and communication interface 3019. The received code may be executed by processor 3005 as it is received, and/or stored in storage device 3011, or other non-volatile storage for later execution. In this manner, computer system 3001 may obtain application code in the form of a carrier wave. Computer system 3001 can transmit notifications and receive data, including program code, through the network(s), network link 3021 and communication interface 3019. --